

Fig. 1

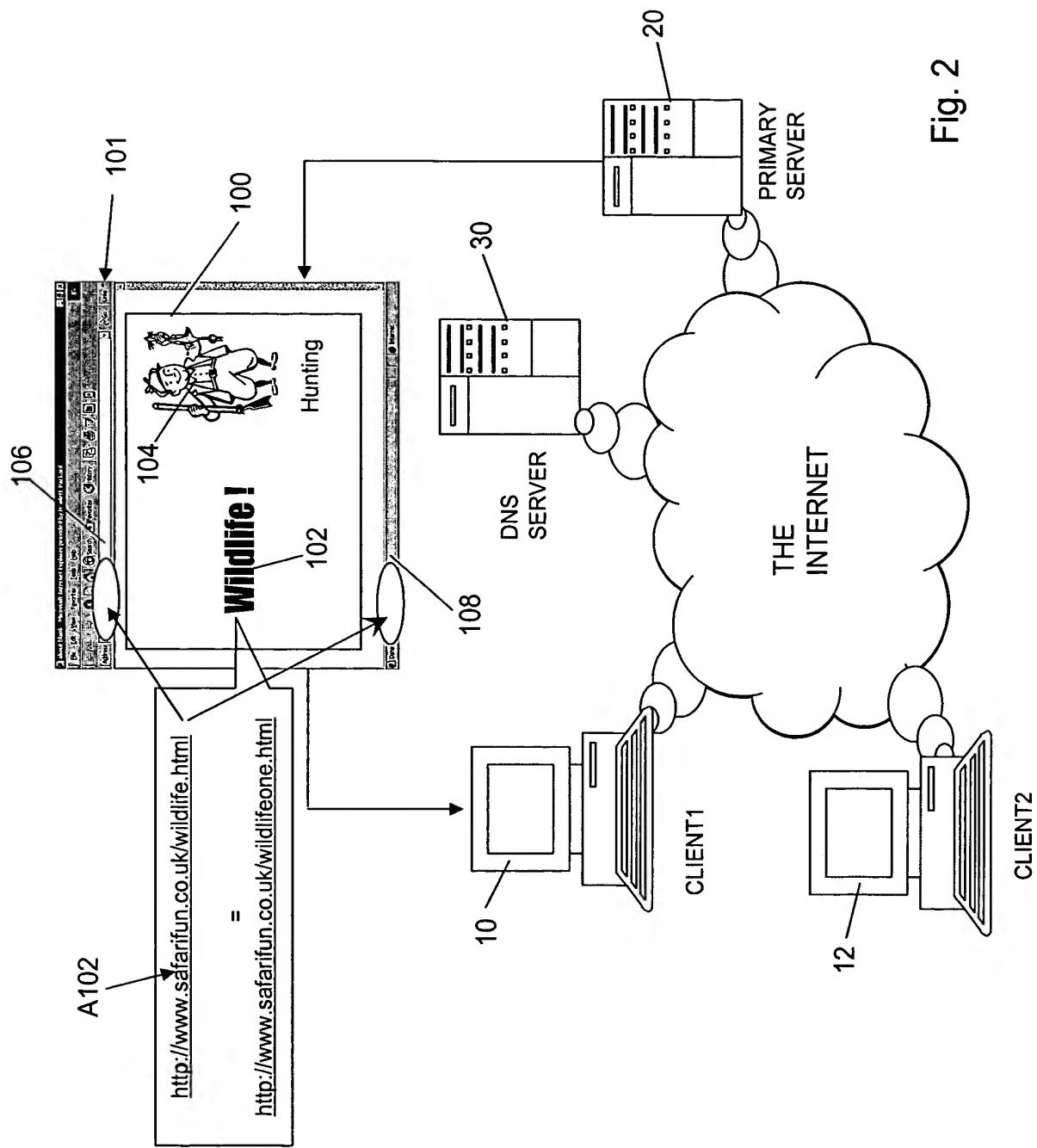


Fig. 2

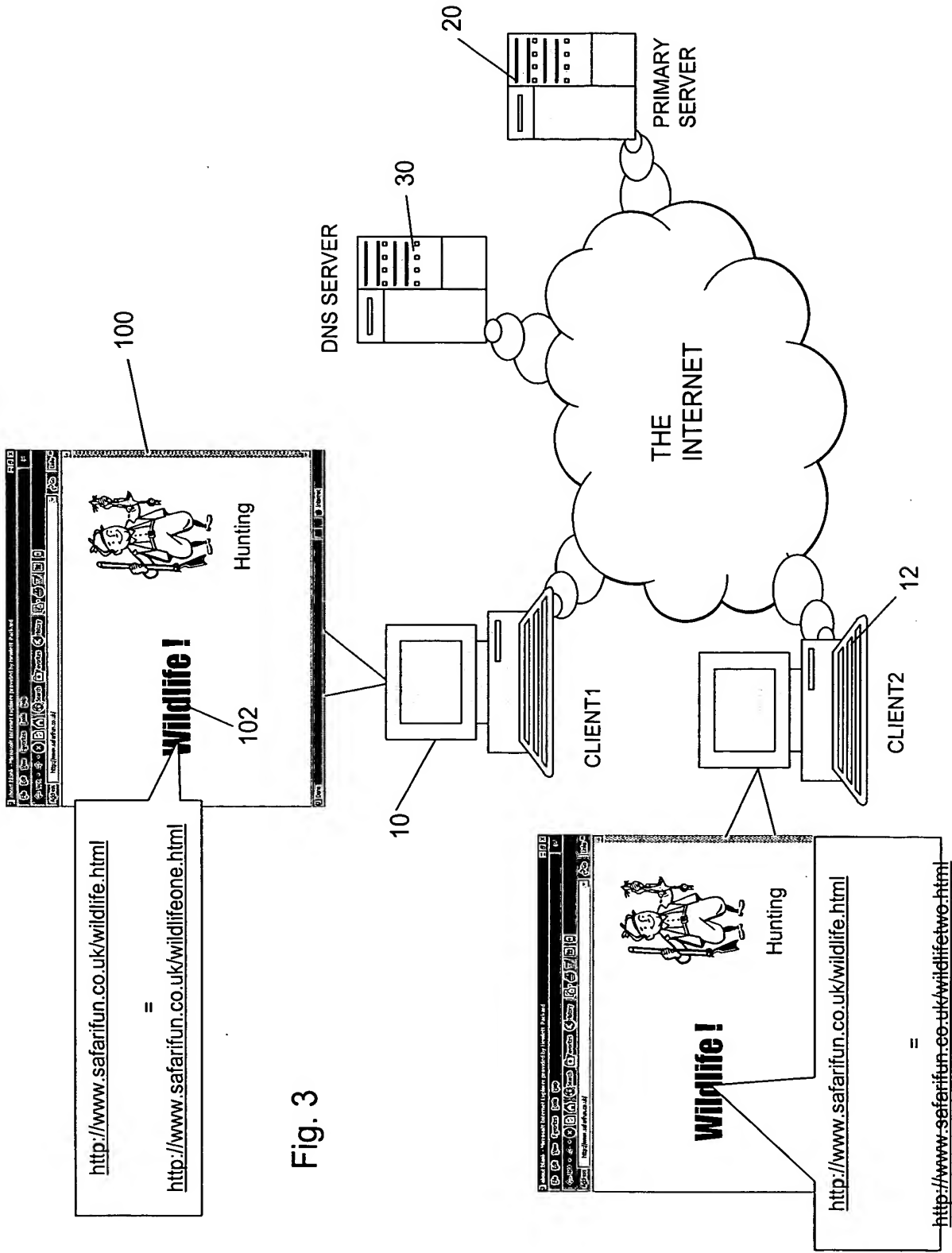
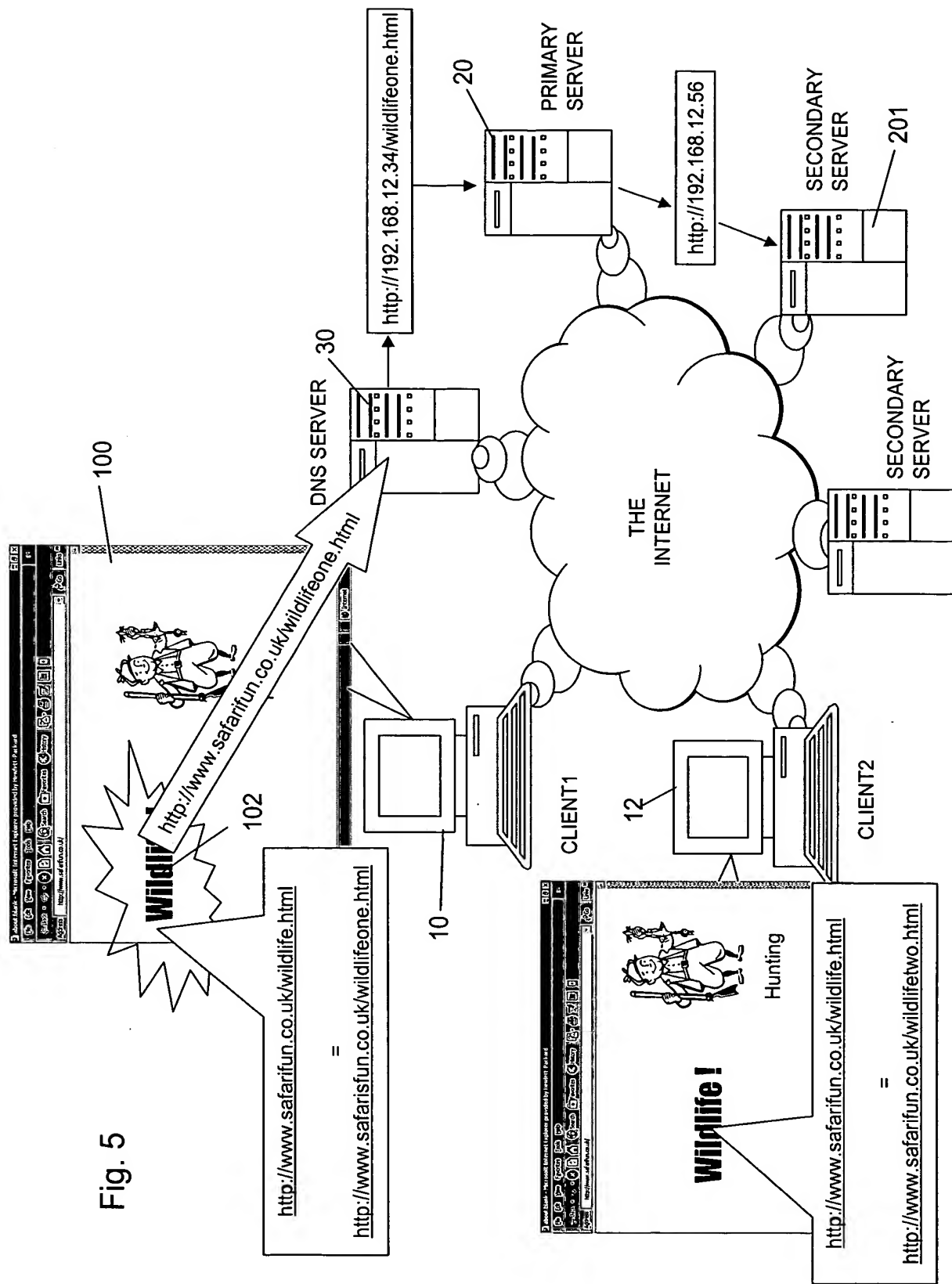
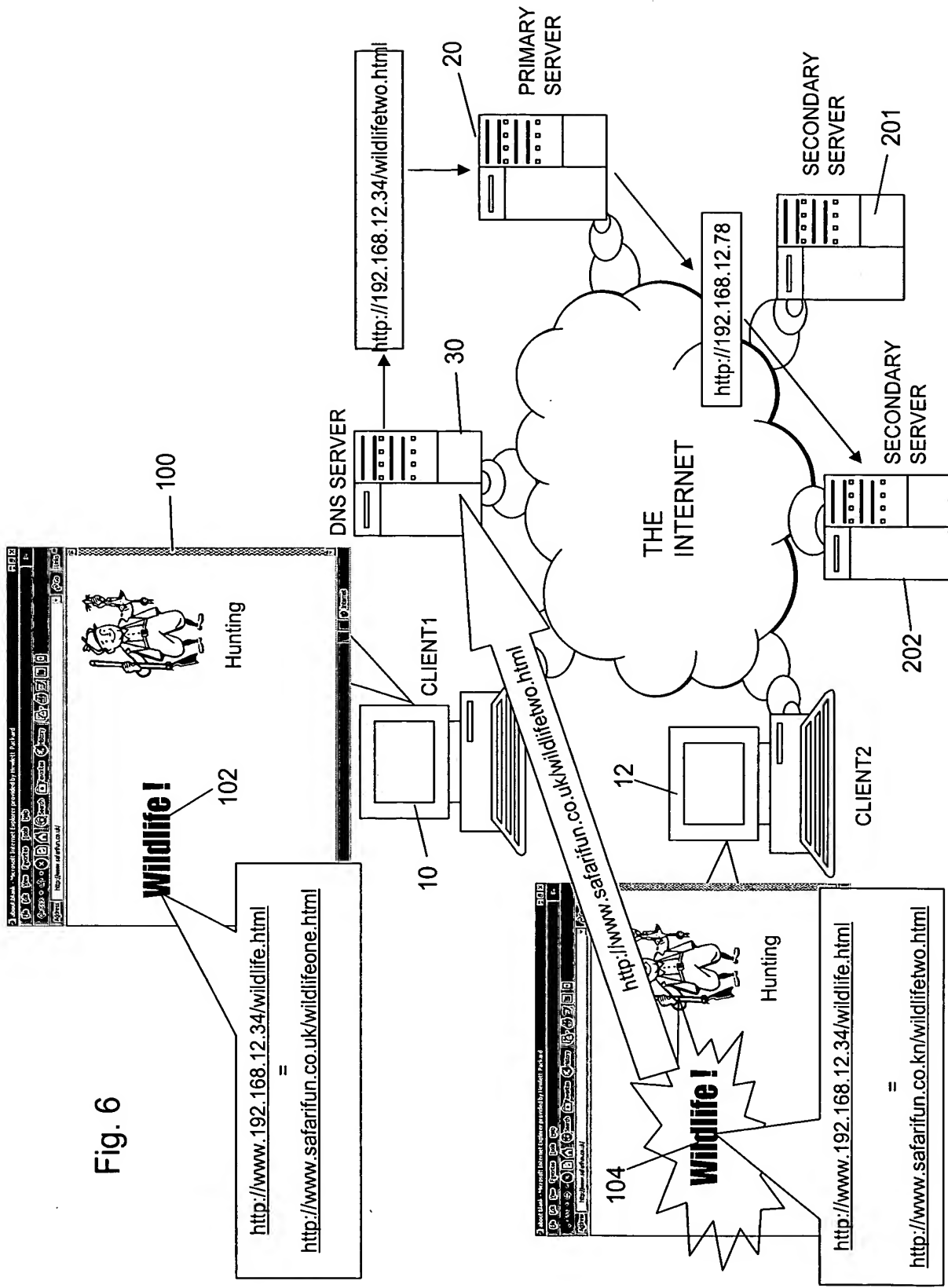


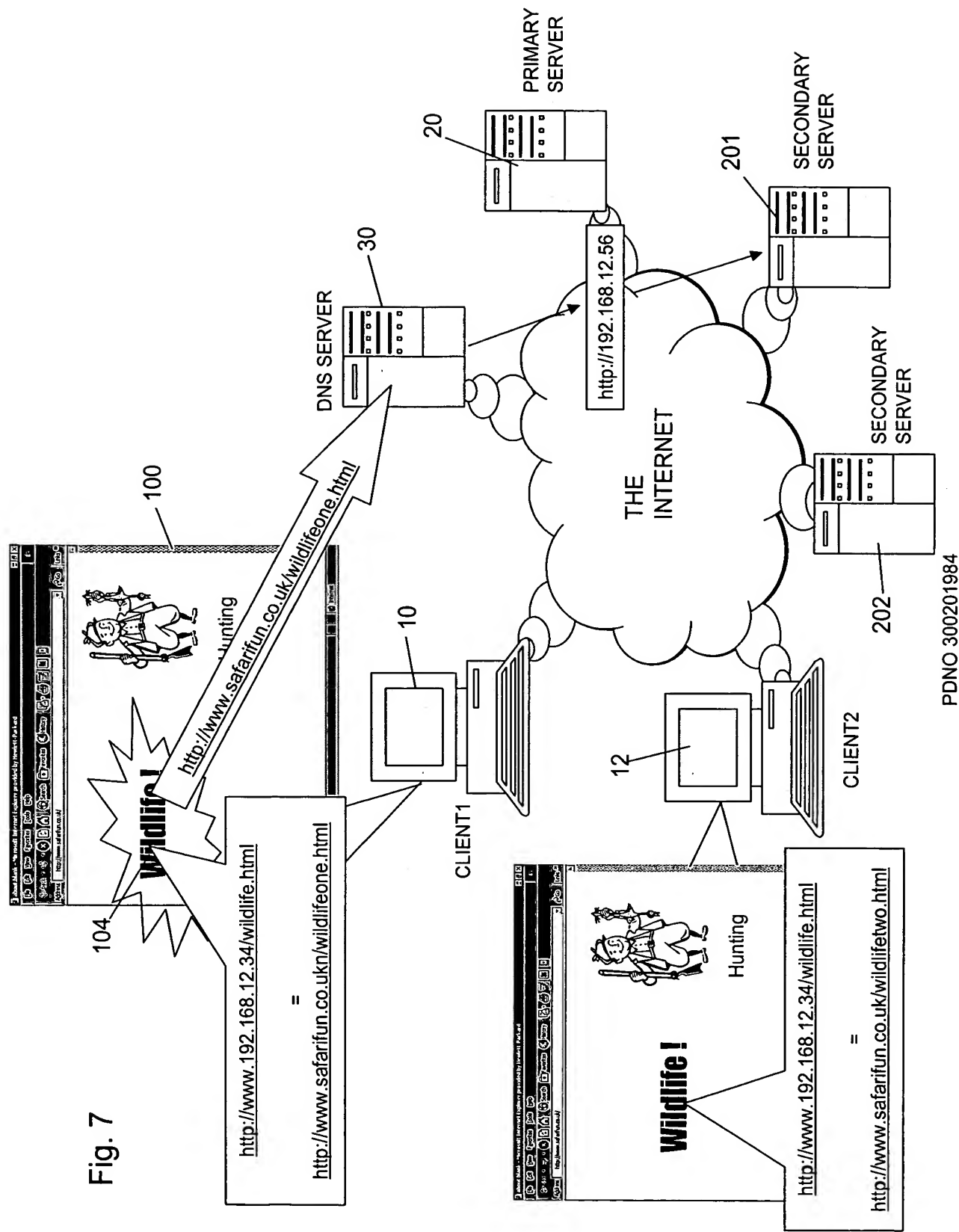
Fig. 3

```
<HTML>
<HEAD>
<TITLE>Safari Fun</TITLE>
<SCRIPT LANGUAGE="JavaScript">
  //
  // a simple alias based upon one common javascript implementation
  // this could also be accomplished through zero height frames with
  // page loaded into upper/lower frame
  function go_to_alias(){
    window.name="Safari Fun";
    window.navigation.bar="www.safarifun.co.uk";
    window.goto("http://www.safarifun/wildlifeone.html");
  }
</SCRIPT>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    // now call go_to_alias
    go_to_alias();
  </SCRIPT>
</BODY>
</HTML>
```

Fig. 4







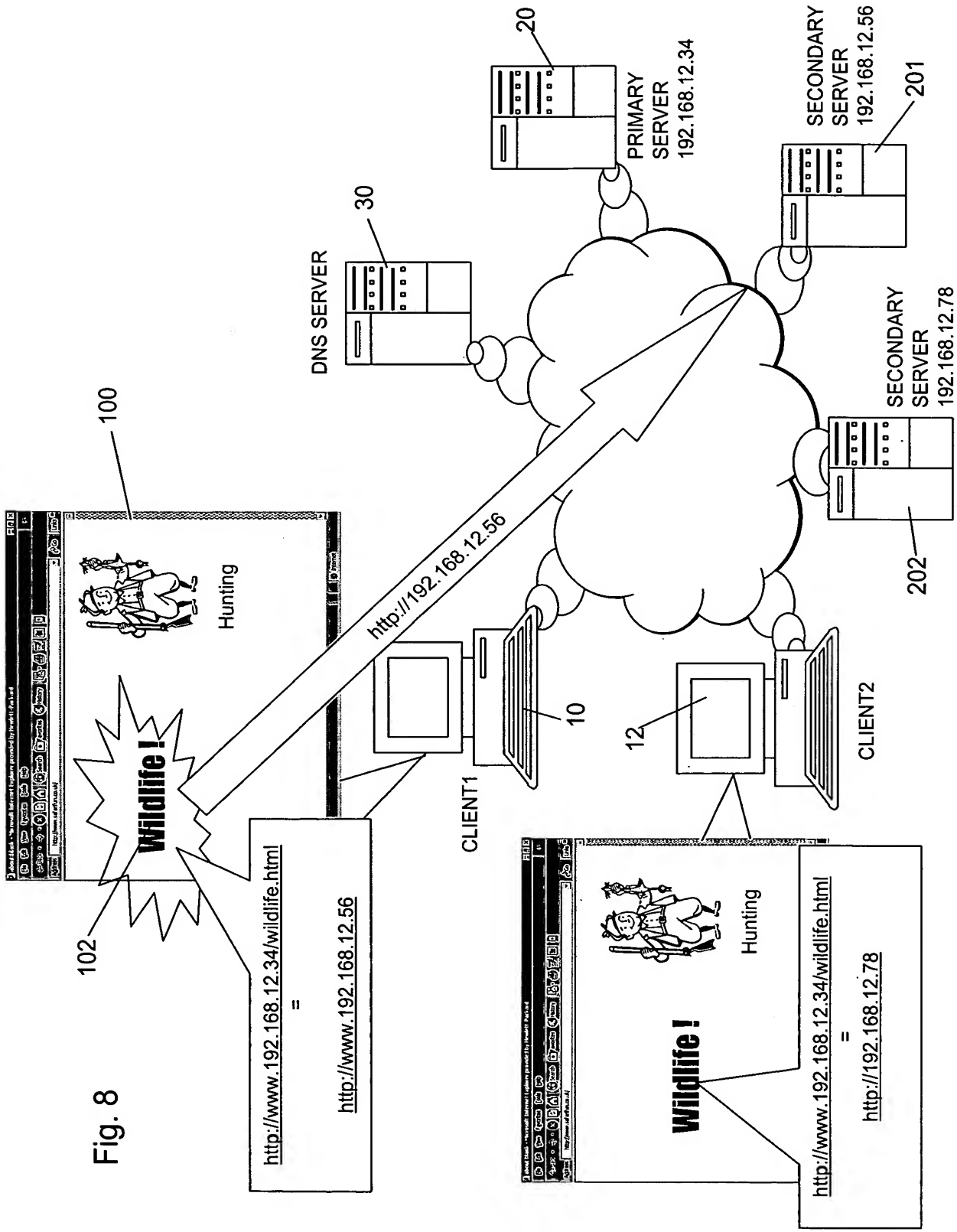


Fig. 8

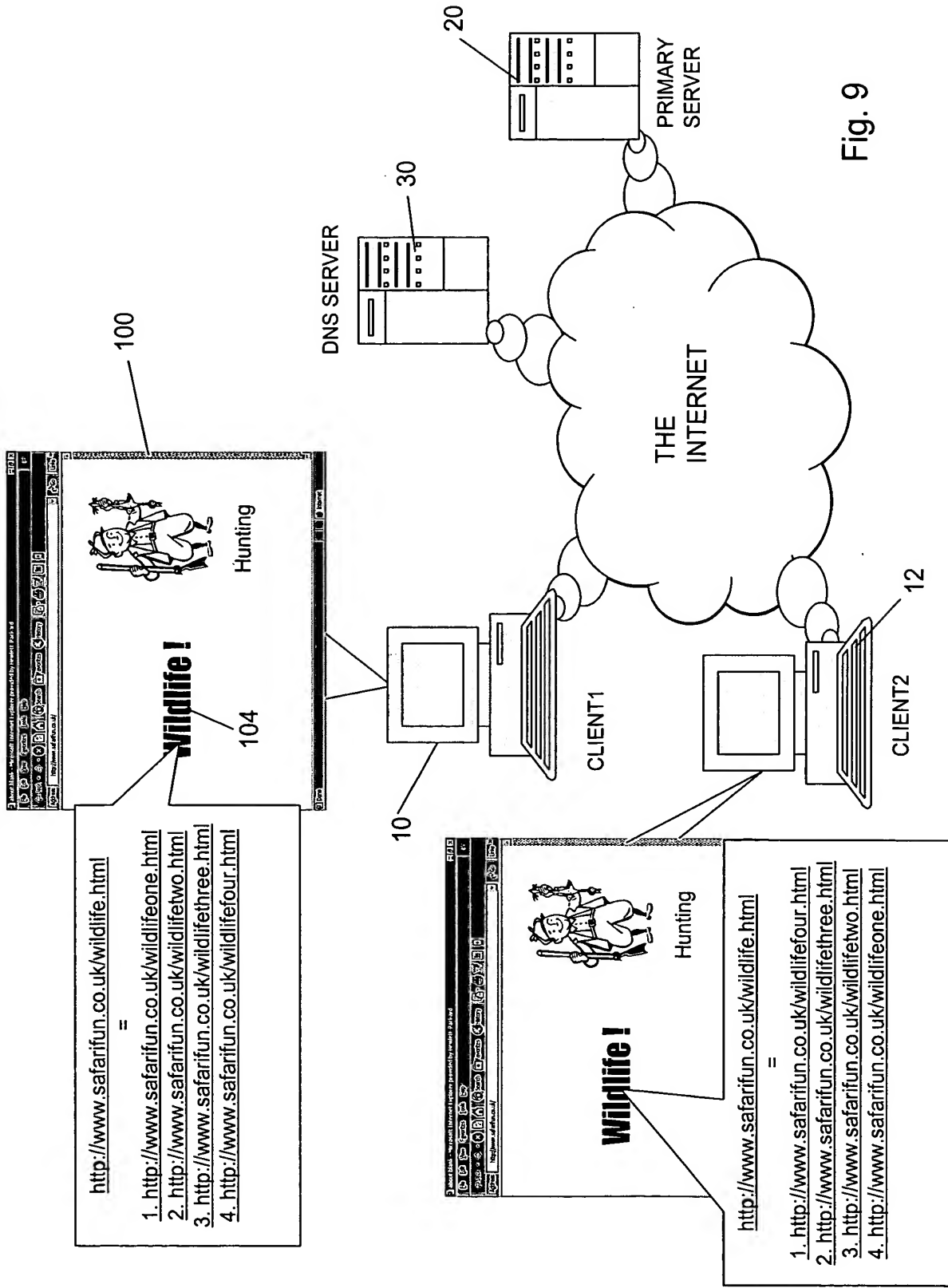


Fig. 9

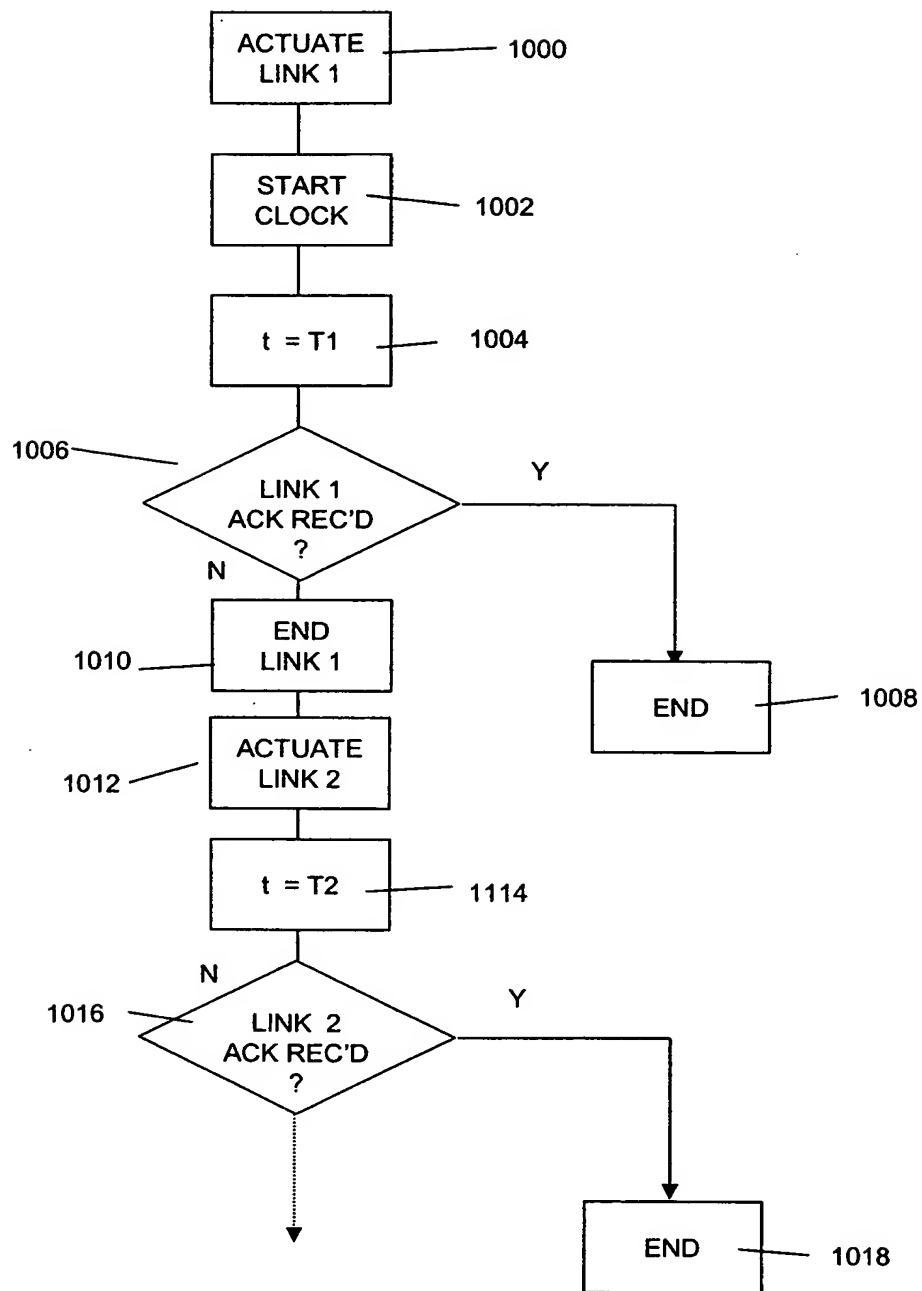


Fig. 10

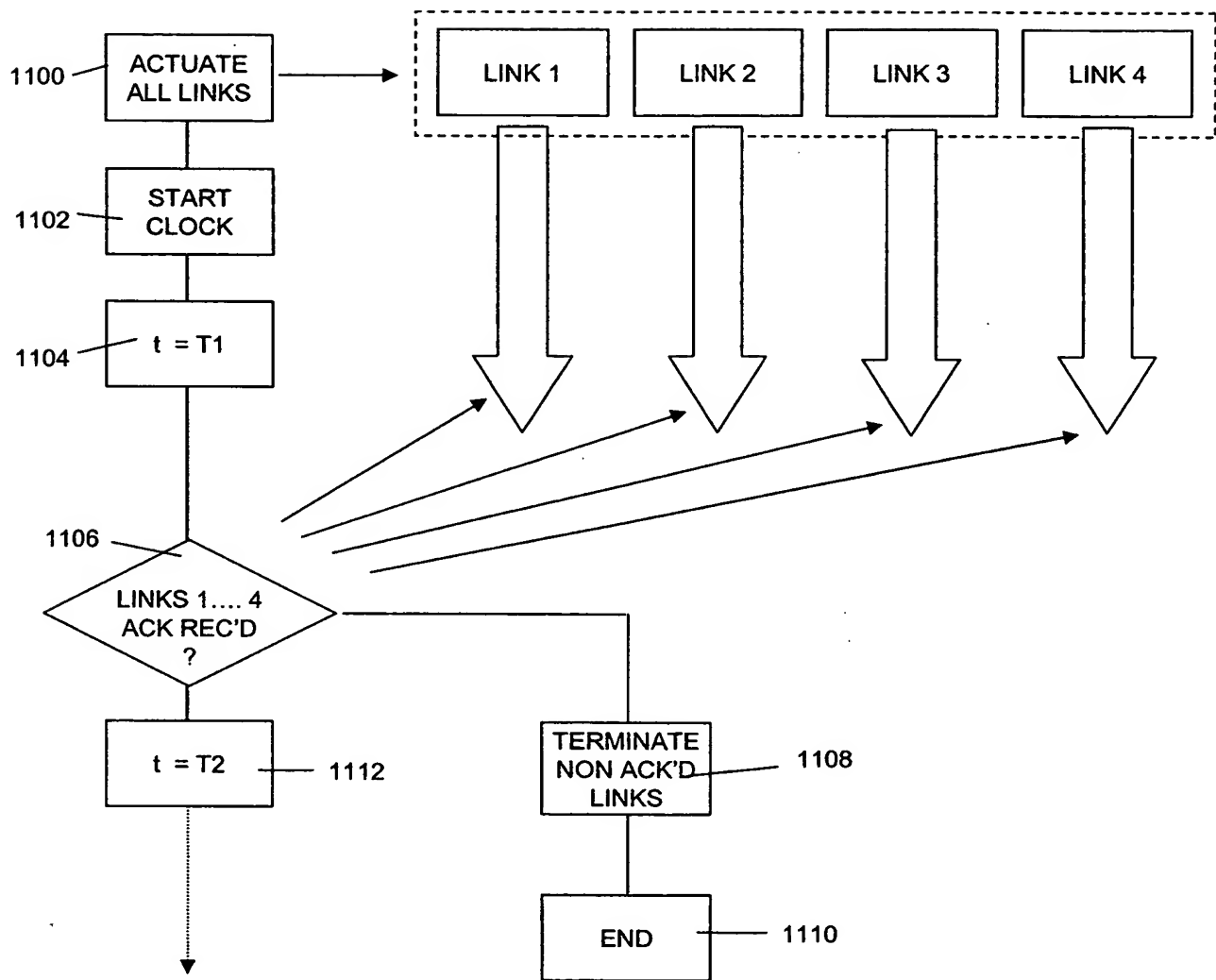


Fig. 11

```

//
// takes as a parameter an array of strings, each of which
// represents a seperate link to try.
//
// executes links in order of presentation and returns successful
// (or none)note : returns string

public class testSLinks
{
    public static String main(String links[], int link_count, int time_out)
    {
        // default return behaviour
        String success_link = "";
        // for each link

        for (int i = 0; i < link_count; i++)
        {
            // fire up a 'getLink' function - not a thread which terminates
            //on either time_out seconds (no response) or
            testThread(link[i], time_out, success_link).start();
        }
        // termination either leaves successful link in success_link
        or blank eturn(success_link);
    }
}

```

Fig. 12

```

//
// takes as a parameter an array of strings, each of which
// represents a separate link to try. Executes an individual
// thread to access each link. One of three things happens:
// no threads complete in time_out; one thread completes
// continues with successful link killing off other threads
// still running; two or more threads complete - effectively a
// link is chosen to be loaded in a non deterministic manner
// (the result of java serialisation), kills off other threads
// still running.
// note: returns string
// note: uses p_threads library to allow continuation as soon
// as one thread completes;

public class testPLinks
{
    public static String main(String links[], int link_count, int time_out)
    {
        // default return behaviour
        String success_link = "";
        // for each link

        for (int i = 0; i < link_count; i++)
        {
            // fire up a 'getLink' thread which terminates on either
            // time_out seconds (no response) or
            new testThread(link[i], time_out, success_link).start();
        }
        // termination either leaves successful link in success_link
        // or blank return(success_link);
    }
}

```

Fig. 13